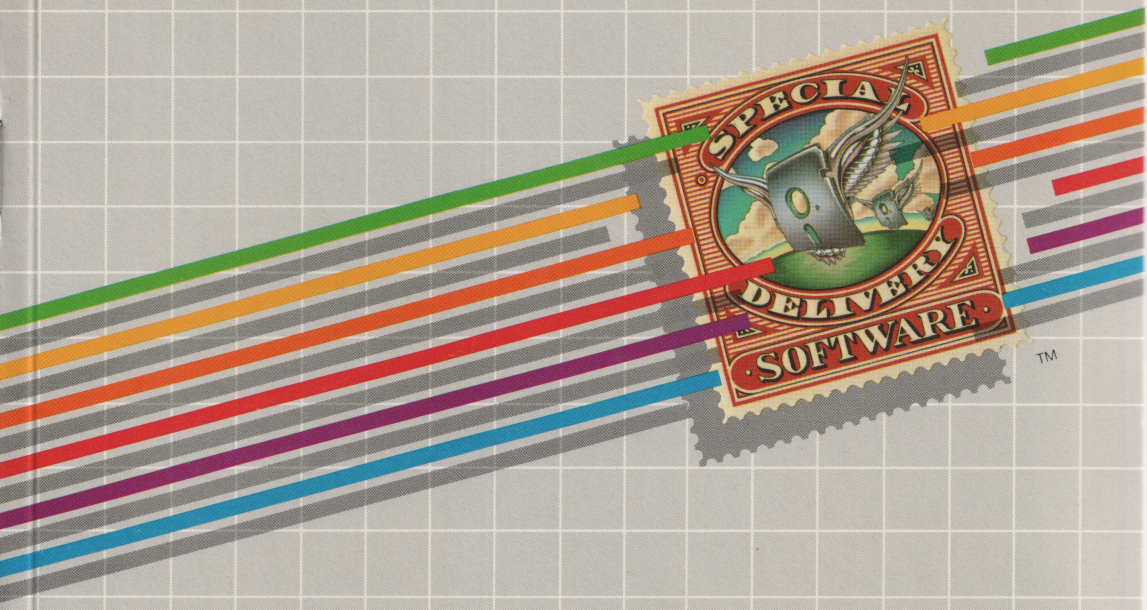
A P P L E



PSORT

Written By: SHATTOCK & ASSOCIATES

In conjunction with Apple Computer, Inc.

NOTICE

Apple Computer Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

APPLE COMPUTER INC. AND SHATTOCK & ASSOCIATES MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. APPLE COMPUTER INC. SOFTWARE IS SOLD OR LICENSED "AS IS." THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT APPLE COMPUTER INC., SHATTOCK & ASSOCIATES, THEIR DISTRIBUTOR, OR THEIR RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION, AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE COMPUTER INC. OR SHATTOCK & ASSOCIATES BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF APPLE COMPUTER INC. OR SHATTOCK & ASSOCIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

1Ø26Ø Bandley Drive
Cupertino, California 95Ø14
(4Ø8) 996-1Ø1Ø

TABLE OF CONTENTS

Introduction

PSORT is a versatile sort/merge program designed for use in microcomputer systems which use the UCSD Pascal system. ("UCSD Pascal" is a trademark of the Regents of the University of California).

The program has the following features:

- Fixed or variable length records.
- Fixed or variable length fields.
- Records can be specified to be included or excluded from the sort or merge processes.
- Up to 10 (user redefinable) sort keys with mixed ascending/descending sequences.
- Merges up to 10 (user redefinable) pre-sorted files.
- Uses either text files, or files of strings.
- Uses tree sort algorithm and dynamic memory allocation.
- Performs "stable" sorting.
- Sort controlling parameters are read from the console or a parameter file.
- Includes a program with thorough error checking to build the parameter file.
- Callable as a procedure from a user program.
- Full Pascal source code (in UCSD Pascal Ver. I.5) provided.

This manual assumes that your Apple II system is correctly set up. If you're not sure that the system is ready to go, see Appendix A.

System Configuration

Use of PSORT assumes that the user has a UCSD Pascal system and at least one floppy disk drive. The full Pascal source code is provided.

The programs are provided in both Pascal source code, and Apple P-code (.CODE) files.

Getting Started

The programs may be modified and recompiled by the user. There are two files which need to be compiled. They are: "PSORT.TEXT", and "SORTPARM.TEXT." "PSORT.TEXT" consists solely of "include file" calls to the four files which make up the sort/merge program. Simply use the filer to G(et PSORT as the current file and C(ompile it. Return to the filer and save the current file (in fact, only the code file will be saved - the text file is already saved).

"SORTPARM.TEXT" is the program which is used to build the parameter file. Compile it also.

Execution of PSORT

PSORT is executed by typing X at the outer level of the Pascal system. Type "PSORT" when the system asks for the file to execute. The program will then announce itself and ask for the name of the parameter file. Enter a <return> if you wish to enter the parameters directly from the console. If you give a file name (don't give the suffix ".TEXT" - it will automatically be appended), the program will attempt to find that file and read it. If the file is successfully found, then the sorting process will commence. If the "inform" flag has been set, then the progress of the sorting will be displayed on the console, and at the end of processing, relevant statistics will be displayed.

Because PSORT uses "segment procedures", the disk containing it must remain on-line during its execution.

Text Files, Files of Strings

PSORT can read or write either standard Pascal text files ("File of char") or files of strings - i.e., the file is defined to be "File of string (<length>)." Files of strings take less time to read and write than do text files, but may involve more disk space. Files of strings are not directly human readable, and are not recommended for the final output file unless that file is to be read by another program. To speed the processing of the intermediate files, files of strings can be used.

Any programs which produce the input to PSORT or read the output file from PSORT as files of strings, must use the same string size declaration as does PSORT.

Fixed, Variable Length Keys

PSORT can sort files whose records consist of either fixed length keys, or variable length keys.

Fixed length keys are those which consist of a fixed number of characters. Each key is referenced by its starting character position and its length. The record may in fact be of variable length, but all fixed length keys must appear before the variable part of the record.

Variable length keys consist of a variable number of characters. Each key is separated by a special character called the "key delimiter." The keys are referenced by their position in the record. It is

permissable to have null keys (i.e., immediately consecutive delimiters). Such keys will be lowest in the collating sequence.

In the distributed version of PSORT, the maximum number of characters in a key (either fixed or variable length) is 25. If a field longer than the allowable maximum is encountered in the record, then that field will be truncated (left hand portion retained) to the allowable maximum. There will not be any message to inform the user that the truncation has occurred.

Selection of Records Using Include/Exclude

PSORT allows the user control over which records are included in the sort (or merge) process. The user may specify whether certain records are to be included in the sort, or whether certain records are to be excluded.

For each field in the record, the user may specify a value which is compared with that particular field in each input record. If the records are being "included", then only records which match the specified selection values will be included in the sorting process.

Conversely, if records are being "excluded", then records which match the selection values are not included in the sorting process.

Any of the fields can be assigned several "selection values." For example, suppose the selection values have been given as follows:

Field	Value
1	Nails
3	Apples
3	Pears

If the "include" mode is being used, then only records which have Field #1 equal to "Nails" or Field #3 equal to "Apples" or "Pears" will be included in the sort. Note that a record will be included (or excluded) if ANY of the fields match (i.e., once one field has matched, no further fields are compared).

These selection features allow PSORT to be used as a primitive data base system. Thus, the user can set up the sort program to effectively obey the command: "List all records having city names of 'Hopeville' or 'Doomtown', or (not "and") surnames of 'SMITH' (regardless of the city name), sorted in descending order of city names, with an ascending subsort on surnames."

Each selection value string must correspond in length to the corresponding field length in the record. No check is made of this, except the SORTPARM program (see later) checks that the value string is not longer than the current maximum allowable key length. The selection string must also agree in case with the field (i.e., no

translation from upper to lower case, or vice versa, is done).

The SORTPARM Program

PSORT obtains its controlling information from either a text field or the console. SORTPARM is used to interactively obtain the various parameters and build the parameter file.

On execution of SORTPARM, the following questions will be asked (the answers can be given in either upper or lower case). Each answer is terminated with a carriage return. Prompts which require only a single character input (plus the carriage return) will accept the ESCAPE character and terminate SORTPARM.

Prompt: Name of parameter file (.TEXT will be added):

Response: Enter the name of the file which will hold the parameters. Any previous file of this name will be removed.

Prompt: S(ort, or M(erge only?

Response: Enter "S" if a sort of a single file is being performed; "M" if a multiple file merge is being performed.

If sorting, the following will then be asked:

Prompt: Input file name:

Response: Enter the full name of the input file to be sorted (including the .TEXT suffix, if a text file).

Prompt: Is it T(ext or F(ile of strings?

Response: Enter the appropriate letter.

Prompt: Output file name:

Response: Enter the full name of the output file (including .TEXT).

Prompt: Is it T(ext or F(ile of strings?

Response: As before, depending on what type of file you wish

the output file to be. If the file is to be directly readable, enter "T."

Prompt: Volume name for intermediate files (or ret):

Response: If you have a two drive system, the intermediate work files can be directed to either disk to provide more space. Terminate the volume name with ":." If <return> is entered, then the current default prefix is used.

Prompt: Are they to be T(ext or F(iles of string?

Response: To speed up the processing, the intermediate files can be written as files of strings rather than text files. If disk space is at a premium, then use text files. This will only save space if the actual record length is significantly less than the program-defined maximum record length ("mrl"). (80 characters in the distribute version).

If merging, then the following is asked:

Prompt: Enter file to be merged. Terminate with "END":

Response: Give the file names of the input files to be merged. The files do not have to reside on the same disk as each other. If the maximum number of file names is reached before the user types "END" (or "end"), then the user will be asked whether he wants to continue entering parameters. Entering "no" will terminate SORTPARM. A "yes" response will perform the merging process using the input files just entered. If there are more than the allowable number to be merged, additional merges will have to be performed.

Unsorted files may be submitted, but the sequence of the output file will be unpredictable.

It also is legitimate to merge only one file - this is equivalent to a copy of that file from the input file to the output file.

Prompt: Are the files T(ext, or F(ile of strings?

Response: Enter the appropriate letter for the merge input files. Note that all of the files must be of the same type.

Prompt: Want to be informed of progress and statistics?

Response: Answering "yes" to this will cause the sort program to indicate its progress, and display statistics at the end of its processing.

The program indicates whether it is in the distributing phase (where it creates any necessary work files); then when it is merging the intermediate workfiles (if any). If no intermediate workfiles are produced, then the message "memory contained" is displayed.

The statistics that are displayed are as follows:

Number of records read.
Number of records written.
Number of merge input files (if merging).
Number of intermediate workfiles created (if sorting).
Number of keys specified.
Number of records on intermediate files (if any).
Buffer space available (this is the space available for the tree sort).

Prompt: Keytype - F(ixed or V(ariable:

Response: If you are using keys which are fixed length (i.e., each corresponding field of each record starts in the same column), enter "F." Variable length keys are those whose fields are separated by a "key delimiter" - usually a special character.

If variable length records -

Prompt: Number of fields.

Response: Enter the number of fields which are in the record. This value is used by SORTPARM to help check the validity of the following entries.

Prompt: Key delimiter:

Response: Enter the character which is being used to delimit the fields. The carriage return at the end of the record will automatically act as a delimiter. (Blank as a key delimiter must be typed as a single space, i.e, the pressing of the carriage return alone is not sufficient to establish blank as a delimiter).

For fixed length keys:

It is necessary that PSORT knows the structure of the fixed length record (i.e., it must be told where each field starts), and its length. The following sequence requests the length of each field, starting from field number 1, the left most. PSORT automatically calculates the field starting positions from the given field lengths. Normally, a field length cannot exceed the current maximum key length. However, if a field is not being used as a key, it can have a length greater than the normal maximum. If such a larger than normal value is entered, SORTPARM will ask you if you meant it. If so, it will be accepted, but that field will not be permitted to be used as a sort key, or as a selection field.

Prompt: Field # <n> length:

Response: Enter the length (in characters) of the appropriate field. Terminate the sequence by entering zero. The distributed version of PSORT allows 30 fields per record. If this number is reached, SORTPARM will request whether you wish to continue. If so, then any further fields in the record will not be usable. If not, SORTPARM will terminate.

The following data is required to determine the order of the sorting. In the distributed version, up to 10 sort keys may be specified. The major key is number 1, and secondary keys are the subsequent ones. Secondary keys have significance when the fields of a more major key are equal.

For both fixed and variable length keys:

Prompt: Key # <n> field number:

Response: Enter the number of the field to be used for the key. Fields are numbered starting from 1, from left to right. Use a 0 to terminate the sequence.

Prompt: Key direction - A(sc or D(esc:

Response: Enter with either "A" or "D" depending on whether the particular field is to be sorted in ascending or descending order, respectively.

Prompt: Want to specify selection fields?

Response: Enter "yes" (or "y") if you wish to specify values

which are to be used to determine whether a record is to be included or excluded. Entering "no" will end the execution of SORTPARM and write the parameter file.

Prompt: I(nclude, or E(xclude?

Response: Enter the appropriate letter.

Prompt: Field number:

Response: This specifies which of the fields is to be assigned the following string value which will be used as the comparison value for either including or excluding the record. Enter zero to terminate the sequence. The distributed version of PSORT allows 25 selection values to be specified.

Prompt: Selection value:

Response: Enter a string which is to be used as the comparison value for the particular field.

Console Input

Rather than execute the SORTPARM program to build the parameter file, the user may enter the parameters at the execution time of PSORT. The same questions as described previously are asked, except for the following:

No parameter file name is requested.
No request is made for the record length (for fixed length fields) or number of fields (for variable length fields).

Collating Sequence

The collating sequence is the ASCII character set. Because only exit fields (or their content-equivalent "file of strings") are read, numbers are also read as characters. This does not cause any difficulty for sorting because, for example, "123" is less than "234", as expected. Because the blank (space) character is less than the digits, "12" is less than "112", also as expected. Hence, numbers should be right justified, blank or zero filled. Decimal numbers should have the decimal points aligned.

Errors

Some error checking is performed by PSORT. If any of the parameter, sort input, or merge input files are not found, an appropriate error message is displayed, and the program terminates.

In the case of the merge input files, all of the files are checked to see if they exist. All that are missing are reported before the program terminates.

If, during sorting, the number of intermediate files exceeds the limit, the job is terminated. The program can be recompiled with a greater limit, or the input file converted to two (or more) files. These files can then be resorted and the resultant sorted files merged.

Any other errors (caused by perhaps invalidated input, or, for example, no disk space) will be detected by the Pascal system, and will result in the whole system being re-initialized.

Capacity

The Apple II based system will be able to sort approximately 1500 records of 80 characters each, which is close to the storage capacity

of a 16 sector disk. This capacity can be increased by changing the maximum number of workfiles (see below), or by using a small line length, if appropriate (see below). Note that this capacity depends (in part) on the amount of memory left free by the host Pascal system, and hence, it is influenced by, for example, the size of the p-code interpreter for your particular processor.

If PSORT is being used as a procedure, its capacity may be less than this because the calling user program will also be using memory, hence, there will be less space available for PSORT's tree.

Apart from disk space, there is no limit on the size of files which are being merged.

How to Change the Distributed Limits

The distributed version of PSORT has the following limits:

- 10 keys
- 10 intermediate workfiles or merge input files
- 80 character length input files
- 25 character key length
- 30 fields in fixed length records
- 25 selection values

All of the limits, except for the number of intermediate files, can be changed simply by editing and recompiling. The absolute upper limit on the line length is 255 (USCD Pascal limit). The line length should not be made too much larger than what is required because the value determines the size of the record which is put on the node of the tree, and hence, uses up memory. This value also affects the disk space used when files are written as "files of strings." Text file records only contain the actual number of characters in the string. The key length limit should not exceed the line length limit.

Apart from a small increase in memory for each additional key allowed, there is no limit on the number of keys.

To change the number of intermediate files is more complicated. The major limitation is the rapid use of memory for each additional file - approximately 300 words (600 bytes) for each file variable. Unfortunately, USCD Pascal does not support arrays of files, hence, each file must be opened and closed explicitly. Part 3 of the source code contains the code for simulating arrays of files. There is code there for up to 20 intermediate files, with those above 10 made inactive (to save memory) by comment delimiters (* and *). These delimiters will have to be relocated by editing to increase the number of files. It is also necessary to shift the delimiters around the inactive file variables in the "var" section of the procedure "merge." Be careful that increasing the number of files does not increase the memory requirements of "merge" beyond your physical memory.

The numeric values to change are defined in the "const" section of Part 1 of the source code.

Also in Part 1 of the source code is the code which sets up the names of the intermediate work files. It is necessary to change the comment delimiters here to match the number of work files being used.

Any such changes should also be reflected in the SORTPARM program so that it can perform the appropriate error checking. That is, change the constants at the beginning of the program.

Sorting Method

PSORT uses a combined memory tree sort and disk merge algorithm. For the "distribute" phase, the input file is read, and a record is put onto the correct lead of the tree. To increase the sorting speed, the major key is also stored on the tree. The comparison method is such that equal keys result in a "stable" sort (i.e., for equal keys the records appear on the output file in the same order as they were on the input file).

If the available memory becomes exhausted and there is more data on the input file, then an intermediate workfile with the name SRTn is created. An in-order traversal of the tree then writes records to this intermediate file. Hence, this file is sorted. This process is repeated until the input file is exhausted.

The "merge" phase then operates. The next logical record is selected from the intermediate workfiles and written to the output file. This is repeated until all intermediate workfiles are exhausted. The selection method also ensures that the sort remains stable.

Sorting Large Files

Even in a two drive system, there may not be enough disk space available to allow the sorting of large files. The amount of disk space needed is approximately three times the size of the input file, made up of the input file itself, the workfiles, and the output file. If the input files take up nearly one disk, the workfiles will also use up about the same space - and hence, will have to be on a disk other than the input disk. Thus, on a two drive system, there is no room left for the output file! However, PSORT allows such large files to be used. The method is as follows:

```
Disk #1 contains the Pascal system and PSORT.CODE.
Disk #2 contains the input file.
Disk #3 contains the output file.
```

Insert Disk #1 in Drive "A" (Unit #4: in USCD Pascal terminology), Disk #2 in Drive "B" (Unit #5:), boot up and eX(ecute PSORT. PSORT

will then perform the "distribute" phase, using Disk #1 to receive the work files. When it is time for the output file to be written, PSORT will ask the user to put in the volume (disk) which is to receive the output file. Put Disk #3 into Drive "B" (#5:, after removing Disk #2), and press <space> to continue.

For this method of operation, the volume name of Disk #1 must be given in reply to SORTPARM's request for the volume name for the intermediate work files. Actually, a <return> will suffice if the default prefix has not been altered since boot time.

It is necessary that the Pascal system remains on-line in case of errors. The PSORT code is also to remain on-line because of the use of segment procedures.

Speed

PSORT is written in Pascal. The object code is interpreted by the host processor. The speed of sorting will not be as fast as an assembly language program. However, the increased reliability, maintainability, versatility, and ease of modification are over-riding factors. The speed can be increased by using "files of strings" (where applicable). Variable length keys may process more slowly than fixed length keys if multiple keys are being made.

The user should be aware that tree sorting methods have a particular disklike for files which are already sorted or inversely sorted. These types of files result in unbalanced trees, and result in longer tree-building times.

If the output of PSORT is to go to the printer, then the time taken for the "merge" phase of the sort can be made effectively transparent by specifying the output files as PRINTER:, rather than as disk files to be later transferred to the printer.

Conversion to Other Versions of Pascal

PSORT uses several features of UCSD Pascal that are not "standard Pascal." These have been used for speed, clarity, or special applicability to microcomputer systems.

Strings and their relevant functions are used extensively. These can be converted by explicit manipulation of packed arrays of characters. The UCSD facility for turning off I/O error checking is also used in places. The "close" function is used to manipulate files. Other systems may use an external job control language to perform file handling (e.g., checking for existence of files).

The "exit" routine is used to terminate the program after fatal errors (only "exit <main program>" is used).

The function "memavail" is used to limit the size of the tree. Versions without this function will need to monitor the size of the tree by other means - possibly by recording the number of nodes that have been put on the tree. This number (of nodes) would be a function of available memory and the record length.

The "mark" and "release" functions are used to recover the memory used by the tree. Other Pascal systems will need to have an equivalent method of releasing memory.

Segment procedures are used to reduce memory requirements by having one-time code in memory only when required. File variables have been defined locally within segment procedures also to save memory.

Use as a Procedure

PSORT can be called as a procedure from a user program. The following method is used to include the necessary source code at compile time. The user program must include the "type" declaration:

```
str = string[25];
```

within its other "type" declarations.

Use the editor to C(opy the file PSORTP (.TEXT is understood) into the user program after its last "var" declaration. This file (PSORTP) contains the include file calls for the source code for PSORT. Note that a single include file comment cannot be used in the user's program because the UCSD Pascal compiler does not allow nested include files.

The procedure is called by:

```
psort(parmname, error, recsread, recswrite);
```

The variables have the following meanings:

parmname: This is the name of the parameter file. It can be null or "console:" for console input.

The other variables all return values to the user program at the completion of the sort.

error: This returns an error code.

0: No errors.
1: Too many intermediate workfiles.
2: File(s) not found.
3: Insufficient key delimiters in

variable length record.

recsread: The number of sort input file records read, or the number of merge input records read, as appropriate.

recswrit: The number of records written to the output file.

Thus, the user program will appear as:

```
program testsort;
const
 ...
type
  str = string[25];

     ...
var
  paramname: str;
  error, recsread, recswrit : integer;
     ...
(*$I PSORTP1.TEXT*)
(*$I PSORTP2.TEXT*)
(*$I PSORTP3.TEXT*)
(*$I PSORTP4.TEXT*)
begin

     ...

  paramname := "paramfile";
  psort(paramname, error, recsread, recswrit);

     ...
end.
```

Special consideration needs to be taken of files which are used by the user program. When a new file is created, the Pascal system allocates the largest space on the disk for that file. If another file is then created without closing the previous file, then the system will respond with "no room on volume." Thus, if the user program has output files open when it calls PSORT, an error message might result. To avoid this, either close all output files before calling PSORT, or use a size parameter when the files are opened by "rewrite."

PSORT sequentially opens and then closes each of its individual workfiles. No files (not even the output file) are left open after the sort. Thus, if the user program wishes to read the output file, it will have to be "reset."

Fatal errors detected by PSORT are displayed on the console. Control then returns to the user program with the appropriate value in "error." It is then up to the user program to decide what further

action to take.

If the "inform" flag is set, then progress messages and statistics will be displayed, as previously described for the stand-alone program.

The procedure could be compiled as a "unit", and the resultant code thus saved in a library. This would avoid the need to compile the PSORT source code each time the user program is compiled. However, there are some restrictions imposed by the UCSD Pascal system. The major one is that "units" cannot include "segment procedures." Thus, all of the PSORT code would be resident in memory at the same time (including the code <or some> of the user program). This may be impossible, depending on the size of your memory. Also, remember that file variables must be in the "interface" part of the unit. The distributed version of PSORT uses files which are local to certain procedures. Any modifications along these lines are left to the user. Consult the UCSD Pascal manual for further information about units.

Parameter File Structure

This section describes the layout of the parameter file. It is a text file which can be created by the editor, a user program, or as is usually done, by the SORTPARM program. Each record (line) is terminated by a carriage return. Except for file names, numbers, and selection value strings, only the first letter of the word is significant. Upper or lower case may be used. The format of the file depends on the particular options that are set. The centralized fields are common. Each record must not include any leading blanks.

```
                    <S(ort or <M(erge


If sorting:
<Input file name>
<T(ext) or <F(ile of strings>


                          <Output file name>
                       <T(ext or <F(ile of strings>


If sorting:                                  If merging:
<Workfile disk name>                         <Merge input file name #1>
<T(ext) of <F(ile of strings>                                ...
                                             <Merge input file # n>
                                             <end> or <END>
                                             <T(ext) or <F(ile of strings>


                             <Y(es) or <N(o>
                          <F(ixed) or <V(ariable) keys


If fixed:                                    If variable:
<field #1 length>                            <number of fields>
      ...                                    <key delimiter>
<field # n length>
      0


                             <key # 1 field no.>
                             <A(scend) or <D(esc>
                                      ...
<key # n field no.>
                             <A(scend) or <D(esc>
                                      0
                    <Y(es) or <N(o>     Selection of records
                                <field number>
                                <string value>
                                      ...
```

```
<field number>
<string value>
0
```

Example

Suppose the following records are to be sorted in ascending zip codes, then, for equal zip codes, in descending surnames:

```
1.......10.......20..................40.........50...
SMITH     John       43 Fifth Ave.        Hopeville  42312
BROWN     Tom        11 Moonlight St.     Doomtown   45051
```

The key type is fixed, with the field lengths as follows: 9, 10, 20, 10, 4.

To use the variable length keys, the data could have been stored on the input file as follows:

```
SMITH,John,43 Fifth Ave.,Hopefille,42312
BROWN,Tom,11 Moonlight St.,Doomtown,45051
```

Here, the key type is "variable", and the number of fields is 5. The key delimiter is the comma (",").

In both cases, key #1's field number is 5, a(scending, and key #2's field number is 1, d(escending.

If all zip codes equal to "40000" are to be excluded from the sort, then the selection value for field 5 would be defined as "40000."

APPENDIX A

SETTING UP THE APPLE II SYSTEM

This appendix includes a list of the equipment you'll need to use the programs on your Apple II. You do not need to read all the manuals, but they should be on hand to answer questions that may arise in operating the equipment (e.g., how to boot a diskette).

In order to be able to provide Special Delivery Software at a lower cost, the disks have been copy protected. This software requires an Autostart ROM, and that the Autostart ROM is the only monitor ROM in the system.

To use PSORT, you'll need the following equipment:

- an Apple II or Apple II Plus with 48K bytes RAM and the Apple Language System;
- an Apple Disk II with Controller (16-Sector PROMs);
- a Video Monitor or Television.

For reference, you should have on hand a copy of the following manuals:

- This Manual (A User's Guide to the Programs);
- an Apple II BASIC Programming Manual (Setting up the Apple II);
- an Apple Pascal Reference Manual.

Putting The Pieces Together

Here are the steps to follow to put your system together:

(1) To set up your Apple II, follow the instructions in the Apple II BASIC Programming Manual. You may not need to attach the Game Controllers, although there is no harm in doing so. Your Apple II must have at least the minimum amount of memory listed under the equipment description for you to use the programs.

(2) If you already have a Disk Operating System, and are using a version of DOS that runs in 13 sectors (DOS 3.2.1 or earlier), you will need to change two proms on your disk controller card to update your system to 16 sectors. Any version of DOS earlier than release 3.3 will need to be updated. These proms are also the same proms that come with the Pascal Language System. Consult a DOS 3.3 manual for these procedures.

APPENDIX B

NOTE FOR USERS OF APPLE PASCAL

The supplied disk is name "PSORT:" and contains the following files:

```
PSORT1.TEXT     - Source code for stand-alone program, Part  1.
PSORT2.TEXT     -   "      "   "       "          "       "   2.
PSORT3.TEXT     -   "      "   "       "          "       "   3.
PSORT4.TEXT     -   "      "   "       "          "       "   4.

SORTPARM.TEXT   - Source code for parameter file building program.

PSORTP1.TEXT    - Source for use as a procedure, Part 1.
PSORTP3.TEXT    -   "      "   "   " "    "         "  3.
PSORTP4.TEXT    -   "      "   "   " "    "         "  4.

(PSORT2.TEXT is identical with the omitted PSORTP2.TEXT).

PSORT.TEXT      - Include file calls for PSORT1..4.
PSORTP.TEXT     -    "      "    "    "  PSORTP1..4.

TESTPSORT.TEXT - Skeleton code showing use of PSORT as a procedure.

PSORT.CODE      - Object code for stand-alone execution of PSORT.

SORTPARM.CODE   - Object code for the parameter file building program.
```

Compilation of PSORT will require that the compiler is in swapping mode, i.e., by using "(*$S+*)" at the start of the program (this is included in PSORT.TEXT and TESTPSORT.TEXT).

All responses to questions MUST be in upper case. Users with lower case facilities may wish to modify the code. Change code of the type "IF INCHAR IN ["N"] THEN ..." to "IF INCHAR IN ["N","n"] THEN ...".

The User Manual outlines a method of increasing the number of intermediate workfiles by simply moving some comment delimiters. The Apple source code does not include the text for the extra intermediate files. Thus, if the number of intermediate workfiles is to be increased, then the extra code would need to be typed in. This situation should not arise, however, because the maximum size of a file which can be sorted using the distributed values will exceed the disk space which is available.

PSORT

C2B0003

652-2024-01

apple computer inc.

SPECIAL DELIVERY SOFTWARE™

CUSTOMER LICENSE AGREEMENT

Dear Customer:

IMPORTANT: The SPECIAL DELIVERY SOFTWARE™ Product that you have just received from APPLE is provided to you subject to the Terms and Conditions of this Software Customer License Agreement. Should you decide that you cannot accept these Terms and Conditions, then you must return your product with all documentation and this License marked "REFUSED" within the 7 day examination period following receipt of the product.

1. License. APPLE hereby grants to you upon your receipt of this product, a nonexclusive license to use the enclosed SPECIAL DELIVERY SOFTWARE™ product subject to the terms and restrictions set forth in this License Agreement.

2. Copyright. SPECIAL DELIVERY SOFTWARE™, including its documentation, is copyrighted by APPLE or, in some cases, by APPLE's software suppliers. You may not copy or otherwise reproduce the SPECIAL DELIVERY SOFTWARE™ or any part of it except as expressly permitted in this License. Any SPECIAL DELIVERY SOFTWARE™ product that is not copy protected may be copied for backup use only, provided that you reproduce all copyright notices and other proprietary legends on such copies.

3. Restrictions on Use and Transfer. The original and any back-up copies of SPECIAL DELIVERY SOFTWARE™ are intended for your personal use in connection with a single computer. You may not distribute copies of, or any part of SPECIAL DELIVERY SOFTWARE™ to others without the specific granting of a Software Distribution License from APPLE for that purpose.

4. Limited Warranty on Media. APPLE warrants the diskettes on which SPECIAL DELIVERY SOFTWARE™ is recorded to be free from defects in materials and faulty workmanship under normal use for a period of 90 days after the date of purchase. If during this 90-day period, a defect in the disk should occur, the diskette and a copy of your receipt may be returned to the SPECIAL DELIVERY SOFTWARE™ Operation at APPLE, and APPLE will replace the diskette without charge. Your sole remedy in the event of a defect in the diskette is limited to the replacement of the diskette as provided above.

5. LIMITATIONS ON WARRANTY AND LIABILITY. EXCEPT AS EXPRESSLY PROVIDED FOR MEDIA, APPLE, ITS SOFTWARE SUPPLIER, DISTRIBUTORS AND DEALERS MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THE SPECIAL DELIVERY SOFTWARE™, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PURPOSE. SPECIAL DELIVERY SOFTWARE™ IS LICENSED SOLELY ON AN "AS IS" BASIS. THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH YOU. SHOULD THE SPECIAL DELIVERY SOFTWARE™ PROVE DEFECTIVE, YOU (AND NOT APPLE, ITS SUPPLIER, DISTRIBUTOR, OR DEALER) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE, ITS SUPPLIER, DISTRIBUTOR, OR DEALER BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF THEY HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

SPECIAL DELIVERY SOFTWARE™
10260 Bandley Drive
Cupertino, CA 95014